Continue

Tts in android

How to use tts in android. Android tts example. Tts contact number. Tts meaning. Tts short for. Tts settings android. Tts examples. Android built in tts. Android tts engine. Tts server android. Tts output in android. How to disable tts in android.

The Android platform has introduced a new feature in version 1.6: Text-To-Speech (TTS), also known as speech synthesis. This allows your device to "speak" text in different languages. Before diving into the TTS API, it's essential to understand some key aspects of the engine. The TTS engine supports various languages, including English, French, German, Italian, and Spanish. Additionally, American and British accents for English are supported depending on the region. The TTS engine requires language-specific resources, such as a voice and dictionary, to function correctly. These resources are not always included with all devices, especially those with limited storage. If a device lacks these resources, the TTS API enables applications to query the platform for availability and initiate downloads. To ensure your application can use the TTS functionality, you should check for the presence of language files using the `ACTION_CHECK_TTS_DATA` intent. Upon successful completion, the TTS engine is ready to be used. If not, the user will need to install the required data by launching the Android Market and initiating the download. Intent installIntent = new Intent(); installIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA); startActivity(installIntent); Context context = this; // pass a reference to the Context to be used TextToSpeech mTts = new TextToSpeech(context, (initStatus) -> { if (initStatus != TextToSpeech.LICENSE_CHECK complete){ // handle the license check status } }, (failure) -> { // handle the failure }); mTts.setLanguage(Locale.US); // load and set the language to English Locale locale = Locale.US; if(mTts.isLanguageAvailable(locale)){ Log.d("Language", "The language is available"); } else { Log.d("Language", "The language is not available"); } Context context = this; mTts.setLanguage(Locale.US); // load and set the language to English String text = "This is a test string in Italian"; mTts.speak(text, TextToSpeech.QUEUE_FLUSH, null); Now that we've properly initialized and configured our application, it's time to make it talk. The simplest way to do so is by using the speak() method. Let's use an example of a talking alarm clock to demonstrate this. First, we need to initialize two strings: "Did you sleep well?" and "I hope so, because it's time to wake up.". We then call the speak() method on each string, passing in TextToSpeech.QUEUE_FLUSH for the first utterance and TextToSpeech.QUEUE_ADD for the second. The TTS engine manages a global queue of all utterances, which are also known as "entries". Each TextToSpeech instance can manage its own queue to control which entry will interrupt the current one. We're using optional parameters to change the playback stream type. For our alarm clock example, we want our text to be played on the AudioManager.STREAM_ALARM stream type so that it respects the user's chosen settings on their device. We do this by passing in a HashMap with the key TextToSpeech.Engine.KEY_PARAM_STREAM and value String.valueOf(AudioManager.STREAM_ALARM). We're also using optional parameters for playback completion callbacks. Since speak() calls are asynchronous, they will return before the text is finished being synthesized and played by Android. If we need to know when a particular utterance is done playing, we can use the TextToSpeech.OnUtteranceCompletedListener interface. In this case, our activity needs to implement this interface. 1. My alarm has been flushed and I have set a new ID for the end of the wake up message. 2. The activity is now notified when the completion of the utterance is implemented in the listener, where if it's equal to the "end of wake up message ID", play annoying music. 3. Rendering and playback while speaking, to record the synthesis to an audio file instead, we can use the speak() method with optional parameters. 4. We can also associate audio resources with speech using other methods, so at this point we have a WAV file that contains the result of the synthesis of "Wake up" in the previously selected language. 5. The TTS instance can be told to associate the contents of the string "Wake up" with an audio resource. 6. Any call to speak() for the same string content as wakeUpText will result in the playback of destFileName, and if the file is missing, it will behave as if the audio file wasn't there. How to Use Google's Text-to-Speech Feature on Android ---------- **Enabling Select to Speak** ---------- To have your phone read texts aloud, follow these steps: 1. Swipe down from the top of the screen and tap the gear icon to open the Settings app. 2. Tap Accessibility > Select to Speak. 3. Toggle the switch to turn it on, then confirm the permissions. **Using Select to Speak** ---------- To hear text read aloud in any app: 1. Open an app and look for the Select to Speak icon on the screen. 2. Tap the icon to trigger Select to Speak. 3. Choose which text to have read aloud by tapping it. 4. Use the playback options to adjust the reading. **Customizing the Voice** ---------- To change the voice used for text-to-speech: 1. Go to Settings > General management > Language and input (or Settings > Languages on some devices). 2. Tap Text-to-speech or Text-to-speech output. 3. Adjust the Speech rate, Pitch, and Language settings as desired. Remember to always shut down the TTS service when done using it. Enjoy the convenience of text-to-speech in your Android apps! Using text-to-speech functionality is seamless when translating languages with Google Lens. Simply point your camera at unfamiliar text, and it will instantly translate it into your native language. Next, select "Speak" to have the translated text read aloud. This feature can be easily toggled off in Settings > Accessibility > Select to Speak. For Android users, this functionality is available within the Google Docs app. However, on a computer, you'll need to download the Screen Reader extension for Chrome first. Once installed, navigate to Tools > Accessibility settings > Turn on Screen Reader Support, then highlight the text and select "Speak" > "Speak selection". To utilize voice typing in Google Docs, place your cursor where you want to start typing, then head to Tools > Voice Typing. Alternatively, use a keyboard shortcut like Ctrl+Shift+S (Windows) or Command+Shift+S (Mac).